A Comparative Study of Floating-Base Space Parameterizations for Agile Whole-Body Motion Planning

Evangelos Tsiatsianas[1,2], Chairi Kiourt[2,3], and Konstantinos Chatzilygeroudis[1,2]

Abstract— **Automatically generating agile whole-body motions for legged and humanoid robots remains a fundamental challenge in robotics. While numerous trajectory optimization approaches have been proposed, there is no clear guideline on how the choice of floating-base space parameterization affects performance, especially for agile behaviors involving complex contact dynamics. In this paper, we present a comparative study of different parameterizations for direct transcription-based trajectory optimization of agile motions in legged systems. We systematically evaluate several common choices under identical optimization settings to ensure a fair comparison. Furthermore, we introduce a novel formulation based on the tangent space of SE(3) for representing the robot's floating-base pose, which, to our knowledge, has not received attention from the literature. This approach enables the use of mature off-the-shelf numerical solvers without requiring specialized manifold optimization techniques. We hope that our experiments and analysis will provide meaningful insights for selecting the appropriate floating-based representation for agile whole-body motion generation.**

I. Introduction

Generating agile whole-body motions for legged and humanoid robots is a fundamental capability for enabling dynamic and versatile behaviors such as jumping, running, or fast reactivity to external disturbances [1], [2], [3]. Trajectory optimization has emerged as a powerful framework for planning such motions by directly computing physically consistent trajectories that respect the robot's dynamics and contact constraints [4], [5], [6]. However, the choice of floating-base space parameterization — that is, how robot floating-base poses are represented during optimization — plays a critical yet often overlooked role in determining the efficiency, robustness, and success of the resulting trajectories. Despite the variety of parameterizations proposed in the literature, there is currently no systematic evaluation guiding the selection of an appropriate representation for agile whole-body motion planning. In this work, we aim to bridge this gap by conducting a comparative study of different floating-base space parameterizations for direct transcription-based trajectory optimization, highlighting their impact on the generation of agile motions in legged systems.

*This work has been partially supported by project MIS 5154714 of the National Recovery and Resilience Plan Greece 2.0 funded by the European Union under the NextGenerationEU Program.

[1]Laboratory of Automation and Robotics (LAR) in the Department of Electrical & Computer Engineering, University of Patras, GR-26504 Patras, Greece, `etsiatsianas@ac.upatras.gr`, `costashatz@upatras.gr`

[2]Archimedes/Athena RC, Greece

[3]Athena - Research and Innovation Center in Information, Communication and Knowledge Technologies, Xanthi, Greece, `chairiq@athenarc.gr`

Prior work has explored various representations such as Euler angles [2], quaternions [7], or full SE(3) transformations [8], but typically focuses on a single formulation without direct comparison to alternatives. It is clear from the literature that operating in the SE(3) manifold directly should yield better solutions and more robust procedures [9], [10], [8], [11]. Nevertheless, approaches that rely on manifold-based formulations often require specialized solvers or custom handling of constraints, which can limit practical deployment [8], [12]. In this manuscript, we make the case that operating in the tangent space of SE(3) allows for the usage of mature and robust numerical solvers (e.g., Ipopt [13]) while maintaining sufficient expressiveness for agile behaviors. This observation motivates our comprehensive evaluation of different floating-base space choices for agile whole-body motion planning in legged systems.

The main contributions of this work are:

- We present a comparative study of different floating-base space parameterizations for direct transcription-based trajectory optimization of agile whole-body motions in legged and humanoid robots.
- We systematically evaluate commonly used representations under identical optimization and numerical settings to ensure fairness.
- We introduce a novel trajectory transcription based on the tangent space of SE(3) for representing the robot's floating-base pose, enabling the use of mature numerical solvers without specialized manifold optimization.

II. Related Work

The choice of the robot's floating-base representation directly influences the transcription of the trajectory optimization problem, which in turn can affect both solution quality and convergence speed. Early methods parameterized orientation via Euler angles, which offer intuitive roll–pitch–yaw decomposition but suffer from poor numerical conditioning when approaching the gimbal-lock singularities. Switching to unit quaternions removes singularities and provides smooth interpolation, yet introduces a non-minimal 4D representation with a unit-norm constraint and a double-cover ambiguity that must be carefully enforced throughout optimization. Operating directly on the SE(3) manifold avoids these issues, but requires specialized manifold-aware solvers to maintain group structure [9], [10], [8].

Despite these varied proposals, prior work typically evaluates only one representation in isolation — Euler angles [2], quaternions [7], or SE(3) [8] — without a head-to-head comparison under identical settings. Consequently, there is

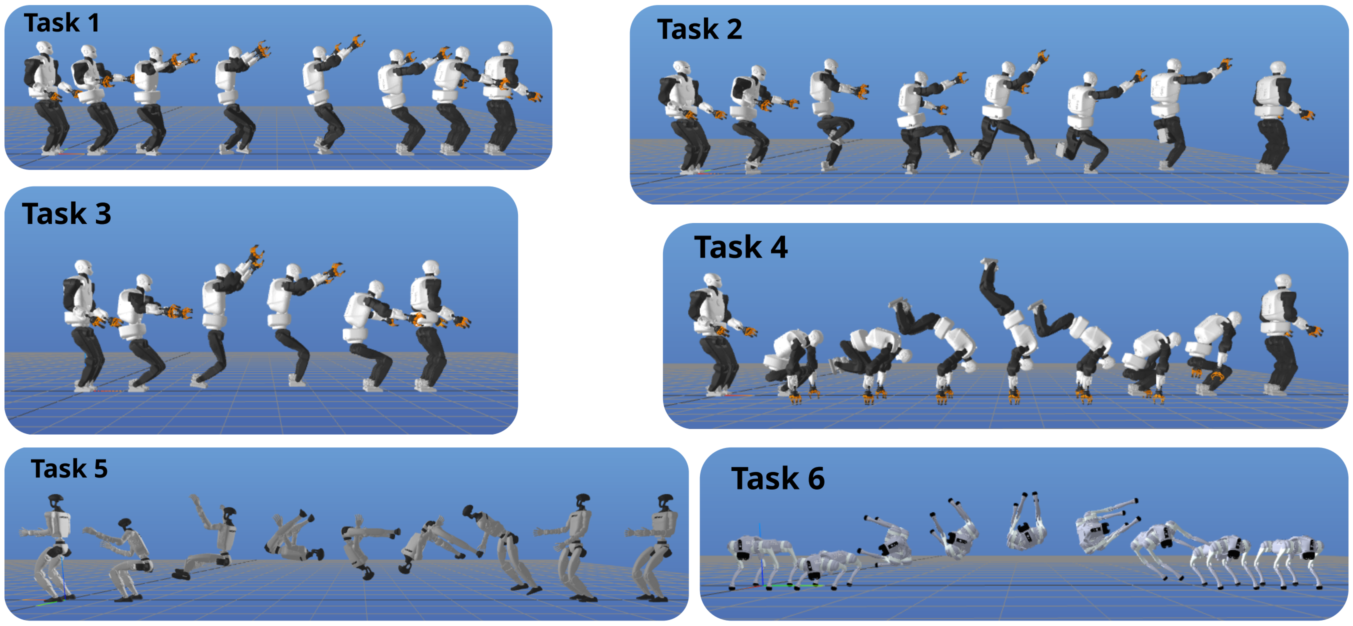


Fig. 1: Overview of the different tasks used for comparisons and indicative solutions. **Task 1 (Walk):** a Talos humanoid walks $2\,\mathrm{m}$ forward. **Task 2 (Hopscotch):** a Talos humanoid is hopping first on the left leg and then on the right ($2\,\mathrm{m}$ forward). **Task 3 (Big jump):** a Talos humanoid is jumping $1\,\mathrm{m}$ forward. **Task 4 (Handstand):** a Talos humanoid is performing a handstand in place. **Task 5 (Humanoid back-flip):** a Unitree G1 humanoid is performing a back-flip (it has to land $0.5\,\mathrm{m}$ backward). **Task 6 (Quadruped side-flip):** a Unitree Go2 quadruped is peforming a side-flip (it has to land $0.3\,\mathrm{m}$ sideways). *For visualization purposes, we have added space between the timesteps*. Please refer to the supplementary video for the real-time visualizations.

no clear guideline for selecting the most suitable floating-base space for agile behaviors involving fast dynamics and complex contacts.

Meanwhile, the numerical optimization community has developed robust, high-performance Non-Linear Programming (NLP) solvers that operate in Euclidean vector spaces, demonstrating reliable convergence across large-scale diverse problems (including robotic applications and trajectory optimization [6]). However, these solvers do not natively support manifold constraints, forcing most implementations to fall back on vectorized parameterizations (e.g. quaternions).

In contrast, manifold optimization techniques for SE(3) and other Lie groups [14] have only recently gained traction in robotics and numerical optimal control. While promising algorithms (e.g., ProxDDP [8] or Riemannian direct methods [12]) show improved theoretical appeal, they rely on custom implementations, and we are yet to see a robust implementation widely adopted and tested by the community. Overall, NLP solvers that operate in manifolds are still in early stage, and do not provide the robustness and effectiveness of the solvers that operate in vector spaces.

In this work, we bridge these two streams by showing that one can enjoy the expressiveness of SE(3) while still leveraging off-the-shelf vector-space NLP solvers. By formulating floating-base poses in the tangent space of SE(3), we maintain manifold consistency without sacrificing solver robustness. We systematically evaluate this approach against Euler Angles and Quaternions for agile whole-body motion planning in legged systems.

III. Background

Planning agile whole-body motions for legged robots is a challenging task, and can be tackled in many ways. In particular, trajectory optimization has emerged as a powerful framework for generating dynamically consistent motions [6], relying heavily on accurate models of whole-body dynamics and suitable representations of the robot's state. This section outlines the mathematical foundations underlying our study.

A. Special Euclidean Group

The Special Euclidean group SE(3) describes rigid-body motions, combining translations and rotations, and is endowed with a Lie group structure. A rigid-body pose $\boldsymbol{T} \in \mathrm{SE}(3)$ is defined as:

$$\boldsymbol{T} = \begin{bmatrix} \boldsymbol{R} & \boldsymbol{p} \\ \boldsymbol{0} & 1 \end{bmatrix} \in \mathbb{R}^{4\times 4}, \tag{1}$$

where $\boldsymbol{R} \in \mathrm{SO}(3)$ is a rotation matrix in the Special Orthogonal group, and $\boldsymbol{p} \in \mathbb{R}^3$ is a translation vector.

The **tangent space at the identity**, denoted $\mathfrak{se}(3)$, is the associated Lie algebra and provides a local linear approximation of the manifold [10], [9]. Although elements of $\mathfrak{se}(3)$ have a non-trivial matrix structure, they can be expressed as linear combinations of basis elements. Hence, they are commonly represented via their vector coordinates, $\boldsymbol{\xi} \in \mathbb{R}^6$.

The exponential and logarithmic maps provide a means to move between the manifold and its tangent space (we use the notation from [10]):

$$\begin{aligned}\mathrm{Exp}(\boldsymbol{\xi}) &= \boldsymbol{T},\\ \mathrm{Log}(\boldsymbol{T}) &= \boldsymbol{\xi}.\end{aligned} \tag{2}$$

Given a small increment $\Delta\boldsymbol{\xi} = \mathcal{V}_b h \in \mathbb{R}^6$ (where $\mathcal{V}_b$ is the body twist and h is the timestep), the retraction operation in $\mathrm{SE}(3)$ is defined as[1]:

$$\boldsymbol{T}_{k+1} = \boldsymbol{T}_k \oplus \Delta\boldsymbol{\xi} = \boldsymbol{T}_k \mathrm{Exp}(\Delta\boldsymbol{\xi}), \tag{3}$$

which ensures that $\boldsymbol{T}_{k+1}$ remains on the manifold. Similarly, we get:

$$\Delta\boldsymbol{\xi} = \boldsymbol{T}_{k+1} \ominus \boldsymbol{T}_k = \mathrm{Log}(\boldsymbol{T}_k^{-1}\boldsymbol{T}_{k+1}). \tag{4}$$

Using the tangent space of $\mathrm{SE}(3)$ enables efficient differentiation [10], [9]. In particular, the left and right Jacobians linearly map local perturbations (e.g., body twists) to variations on the manifold. These Jacobians obey the chain rule for Lie groups [10] and are crucial for computing gradients during trajectory optimization. Ultimately, the tangent space representation of $\mathrm{SE}(3)$ provides a geometrically consistent and computationally efficient framework for the floating-based representation.

B. Euler Angles

An alternative to $\mathrm{SE}(3)$ is the use of Euler Angles for encoding orientation, and a vector $\boldsymbol{p} \in \mathbb{R}^3$ for the translation. We define the *roll* ϕ, *pitch* θ, and *yaw* ψ as successive rotations about the body-fixed axes x, y, and z, respectively. We denote this as $\boldsymbol{\theta} = \begin{bmatrix}\phi & \theta & \psi\end{bmatrix}^\top \in \mathbb{R}^3$. The complete rotation from the inertial/world frame to the body frame is:

$$\boldsymbol{R}(\phi,\theta,\psi) \;=\; \boldsymbol{R}_z(\psi)\,\boldsymbol{R}_y(\theta)\,\boldsymbol{R}_x(\phi), \tag{5}$$

where $\boldsymbol{R}_z, \boldsymbol{R}_y, \boldsymbol{R}_x$ define the rotation matrices around the principle axes. We can relate an angular velocity $\boldsymbol{\omega}_b \in \mathbb{R}^3$ expressed in the body frame, with the roll-pitch-yaw parameters time derivatives as follows:

$$\dot{\boldsymbol{\theta}} = \underbrace{\begin{bmatrix}1 & \sin\phi\tan\theta & \cos\phi\tan\theta\\ 0 & \cos\phi & -\sin\phi\\ 0 & \sin\phi/\cos\theta & \cos\phi/\cos\theta\end{bmatrix}}_{\boldsymbol{W}(\boldsymbol{\theta})}\boldsymbol{\omega}_b. \tag{6}$$

This formulation enables integration using standard numerical schemes (e.g., Euler or Runge-Kutta). However, using this representation, singularities occur when $\theta = \pm 90°$, known as *gimbal lock*. Overall, this representation offers an intuitive, human-readable description of orientation by directly relating rotations to familiar notions of roll, pitch, and yaw, but singularities can be difficult to handle generically.

[1]We use k as the time (or knot point) index.

C. Quaternions

Another alternative representation is the use of unit quaternions for encoding orientation, and a vector for translation. Unit quaternions $\boldsymbol{\rho} \in \mathbb{H}$ form a globally non-singular, smooth representation of 3D rotations. We denote a quaternion as:

$$\boldsymbol{\rho} = \begin{bmatrix}s\\ \boldsymbol{\nu}\end{bmatrix} = \begin{bmatrix}\cos\frac{\theta}{2}\\ \boldsymbol{r}\sin\frac{\theta}{2}\end{bmatrix} \in \mathbb{R}^4, \tag{7}$$

where $\boldsymbol{\nu} \in \mathbb{R}^3$ is the vector part, $s \in \mathbb{R}$ is the scalar part, and $\boldsymbol{r}$ and θ define the axis and angle of rotation, respectively. From Eq. (7) it follows that a quaternion that describes a rotation needs to be of unit norm, $\|\boldsymbol{\rho}\| = 1$.

We can relate an angular velocity $\boldsymbol{\omega}_b \in \mathbb{R}^3$ expressed in the body frame, with the quaternion parameters time derivatives as follows [7]:

$$\dot{\boldsymbol{\rho}} = \frac{1}{2}\boldsymbol{L}(\boldsymbol{\rho})\boldsymbol{H}\boldsymbol{\omega}_b, \tag{8}$$

where

$$\boldsymbol{L}(\boldsymbol{\rho}) = \begin{bmatrix}s & -\boldsymbol{\nu}^\top\\ \boldsymbol{\nu} & s\boldsymbol{I}_3 + \hat{\boldsymbol{\nu}}\end{bmatrix}, \qquad \boldsymbol{H} = \begin{bmatrix}\boldsymbol{0}_{1\times 3}\\ \boldsymbol{I}_3\end{bmatrix}.$$

This formulation enables quaternion integration using standard numerical schemes. Quaternions eliminate singularities and enable smooth interpolation, avoiding issues such as gimbal lock. Nonetheless, their non-minimal 4D representation introduces a unit-norm constraint (or normalization[2]) that must be enforced throughout optimization. Additionally, the double-cover property of quaternions (i.e., $\boldsymbol{\rho} \equiv -\boldsymbol{\rho}$) requires careful handling to maintain derivative consistency in the objective and constraint functions.

D. Optimal Control Problem & Trajectory optimization

The generic continuous-time optimal control problem is given by:

$$\begin{aligned}\min_{\boldsymbol{x}(t),\boldsymbol{u}(t)} \quad & \mathcal{J}(x(t),u(t)) = \int_{t_0}^{t_f} \ell(\boldsymbol{x}(t),\boldsymbol{u}(t))\,dt + \ell_f(\boldsymbol{x}(t_f))\\ \text{s.t.} \quad & \dot{\boldsymbol{x}}(t) = f(\boldsymbol{x}(t),\boldsymbol{u}(t))\end{aligned}$$

where:

- $\boldsymbol{x}(t), \boldsymbol{u}(t)$ are the state and control trajectories,
- $\dot{\boldsymbol{x}}(t)$ the state's time derivative
- ℓ is the "stage cost" with ℓ_f defining the "terminal cost".

The above definition leads to an infinite dimension problem. There are several methods for transcribing it into a nonlinear program of finite dimension suitable for numerical optimization. In this work, we employ *inverse dynamics direct transcription* [15], where the continuous-time problem is discretized as:

$$\begin{aligned}\min_{\boldsymbol{x}_{0:N},\dot{\boldsymbol{x}}_{0:N-1},\boldsymbol{u}_{0:N-1}} \quad & J = \sum_{k=0}^{N-1}\boldsymbol{\ell}(\boldsymbol{x}_k,\boldsymbol{u}_k) + \boldsymbol{\ell}_f(\boldsymbol{x}_N) \qquad (9)\\ \text{s.t.} \quad & \dot{\boldsymbol{x}}_k = f(\boldsymbol{x}_k,\boldsymbol{u}_k)\\ & \boldsymbol{x}_{k+1} = \text{integrate}(\boldsymbol{x}_k,\dot{\boldsymbol{x}}_k)\end{aligned}$$

[2]In our implementation, we always normalize the quaternions and take the appropriate derivatives.

E. Whole-Body Dynamics Formulation

We model robots as free-floating multibody systems, composed of a rigid base and n actuated joints[3]. The generalized configuration of the robot is denoted by $\boldsymbol{q} \in \mathcal{Q} = \mathcal{G} \times \mathbb{R}^n$, where $\mathcal{G}$ encodes the pose of the floating base, and $\mathbb{R}^n$ corresponds to the internal joint configuration. The generalized velocities of the robot are denoted by $\boldsymbol{v} \in \mathcal{H} = \mathcal{V} \times \mathbb{R}^n$, where $\mathcal{V}$ encodes the spatial twist of the floating-base [16], [17], and $\mathbb{R}^n$ corresponds to the velocities of the joints. The system dynamics are described by the manipulator equation [16]:

$$\boldsymbol{M}(\boldsymbol{q})\dot{\boldsymbol{v}} + \boldsymbol{C}(\boldsymbol{q}, \boldsymbol{v}) = \boldsymbol{S}\boldsymbol{\tau} + \sum_{i=1}^{k} \boldsymbol{J}_i^\top(\boldsymbol{q})\boldsymbol{\lambda}_i, \qquad (10)$$

where $\boldsymbol{M}(\boldsymbol{q})$ denotes the symmetric positive-definite generalized inertia matrix. The term $\boldsymbol{C}(\boldsymbol{q}, \boldsymbol{v})$ represents Coriolis, centrifugal effects and gravitational forces. The actuation selection matrix $\boldsymbol{S} = \begin{bmatrix}\boldsymbol{0} & \boldsymbol{I}\end{bmatrix}^\top$ maps joint torques $\boldsymbol{\tau} \in \mathbb{R}^n$ into the full configuration space. Contact interactions are described via the Jacobian matrices $\boldsymbol{J}_i(\boldsymbol{q})$, which relate joint space to spatial velocities at contact points, and the corresponding contact wrenches $\boldsymbol{\lambda}_i \in \mathbb{R}^6$.

The choice of representation for the floating-base configuration directly influences the dimension and structure of the configuration space $\mathcal{G}$, and consequently the generalized coordinate vector $\boldsymbol{q}$. In this manuscript, we consider three alternative representations. The first is the tangent space of the Special Euclidean group at the identity, $\mathcal{G} = \mathfrak{se}(3)$, which provides a minimal, locally linearized representation of the base pose. The second uses unit quaternions for orientation, combined with Euclidean coordinates for translation, yielding $\mathcal{G} = \mathbb{H} \times \mathbb{R}^3$. Lastly, we consider using Euler-Angles for (absolute) orientation along with Euclidean coordinates for translation, yielding $\mathcal{G} = \mathbb{R}^6$. Each of these choices impacts both the formulation of the equations of motion and the numerical properties of the trajectory optimization.

IV. Trajectory Optimization Formulations

To compute dynamically consistent whole-body motions, we formulate a nonlinear trajectory optimization problem by transcribing the system's differential equations into algebraic constraints using direct collocation. The optimization yields trajectories that respect the robot's full-body dynamics, contact feasibility, and kinematic constraints.

The decision variables at each discretization node $k = 0, \dots, N$ include the robot's generalized configuration $\boldsymbol{q}_k$, generalized velocities $\boldsymbol{v}_k$, and accelerations $\dot{\boldsymbol{v}}_k$, as well as the contact forces (and not wrenches[4]) $\boldsymbol{\lambda}_{j,k} \in \mathbb{R}^3$ and contact positions $\boldsymbol{c}_{j,k} \in \mathbb{R}^3$ for each foot j in contact at time step k.

The contact schedule is predefined and encoded via the index set $\mathcal{C}_k \subseteq \{1, \dots, n_f\}$, which indicates the frames in contact at each stage. The integration time step h is kept constant throughout the horizon. Preliminary attempts to include h as a decision variable did not improve performance or solution quality, and are therefore omitted in the final formulation.

The transcribed numerical optimization is as follows:

$$\underset{\substack{\boldsymbol{q}_k, \boldsymbol{v}_k, \dot{\boldsymbol{v}}_k \\ \boldsymbol{\lambda}_{j,k}, \boldsymbol{c}_{j,k}}}{\text{argmin}}\ J = \sum_{k=0}^{N-1} \ell(\boldsymbol{q}_k, \boldsymbol{v}_k, \dot{\boldsymbol{v}}_k, [\boldsymbol{\lambda}]_k) + \ell_f(\boldsymbol{q}_N, \boldsymbol{v}_N) \qquad (11)$$

$$\text{s.t.} \quad \begin{bmatrix}\boldsymbol{0}_6 \\ \boldsymbol{\tau}_{\min}\end{bmatrix} \leq \boldsymbol{\tau}(\boldsymbol{q}_k, \boldsymbol{v}_k, \dot{\boldsymbol{v}}_k, [\boldsymbol{\lambda}]_k) \leq \begin{bmatrix}\boldsymbol{0}_6 \\ \boldsymbol{\tau}_{\max}\end{bmatrix} \qquad (12)$$

$$\text{integrate}(\boldsymbol{q}_k, \boldsymbol{v}_k, \dot{\boldsymbol{v}}_k, \boldsymbol{q}_{k+1}, \boldsymbol{v}_{k+1}, h) = \boldsymbol{0} \qquad (13)$$

$$\text{FK}_j(\boldsymbol{q}_k) = c_{j,k} \quad j \in \mathcal{C}_k \qquad (14)$$

$$\dot{\boldsymbol{c}}_{j,k} = 0 \qquad (15)$$

$$\boldsymbol{c}_{j,k} \in \text{Contact Region} \qquad (16)$$

$$\boldsymbol{q}_k \in \mathcal{Q}, \quad \boldsymbol{v}_k \in \mathcal{H} \qquad (17)$$

$$\boldsymbol{\lambda}_{j,k} \in \Lambda \qquad (18)$$

where $\boldsymbol{\tau}(\boldsymbol{q}_k, \boldsymbol{v}_k, \dot{\boldsymbol{v}}_k, [\boldsymbol{\lambda}]_k)$ refers to solving Eq. (10) for $\boldsymbol{\tau}$, $\boldsymbol{\tau}_{\text{min}}$ and $\boldsymbol{\tau}_{\text{max}}$ are the joint torque limits, integrate$(\cdot, \cdot, \cdot, \cdot, \cdot, \cdot)$ defines a numerical integration scheme, $\text{FK}_j(\cdot)$ performs forward kinematics for the j-th frame, and Λ define the linearized (pyramid) friction cone constraints [6], [18].

TABLE I: Floating-base Space Representations and Operations

Name	Variables	Differences	Elementary Integration
SE(3) Tangent	$\boldsymbol{\xi}_k$	$\text{Exp}(\boldsymbol{\xi}_2) \ominus \text{Exp}(\boldsymbol{\xi}_1)$	$\text{Log}(\text{Exp}(\boldsymbol{\xi}) \oplus \mathcal{V}_b h)$
Quaternions #1	$\begin{bmatrix}\boldsymbol{p}_k \\ \boldsymbol{\rho}_k\end{bmatrix}$	$\begin{bmatrix}\boldsymbol{p}_2 - \boldsymbol{p}_1 \\ \boldsymbol{\rho}_2 - \boldsymbol{\rho}_1\end{bmatrix}$	$\begin{bmatrix}\boldsymbol{p} + \dot{\boldsymbol{p}}h \\ \boldsymbol{\rho} + \frac{1}{2}\boldsymbol{L}(\boldsymbol{\rho})\boldsymbol{H}\boldsymbol{\omega}_b h\end{bmatrix}$
Quaternions #2	$\begin{bmatrix}\boldsymbol{p}_k \\ \boldsymbol{\rho}_k\end{bmatrix}$	$\begin{bmatrix}\boldsymbol{p}_2 - \boldsymbol{p}_1 \\ \boldsymbol{\rho}_2 - \boldsymbol{\rho}_1\end{bmatrix}$	$\text{Log}_q(\text{Exp}_q(\begin{bmatrix}\boldsymbol{p} \\ \boldsymbol{\rho}\end{bmatrix}) \oplus \mathcal{V}_b h)$
Quaternions #3	$\begin{bmatrix}\boldsymbol{p}_k \\ \boldsymbol{\rho}_k\end{bmatrix}$	$\text{Exp}_q(\begin{bmatrix}\boldsymbol{p}_2 \\ \boldsymbol{\rho}_2\end{bmatrix}) \ominus \text{Exp}_q(\begin{bmatrix}\boldsymbol{p}_1 \\ \boldsymbol{\rho}_1\end{bmatrix})$	$\text{Log}_q(\text{Exp}_q(\begin{bmatrix}\boldsymbol{p} \\ \boldsymbol{\rho}\end{bmatrix}) \oplus \mathcal{V}_b h)$
RPY	$\begin{bmatrix}\boldsymbol{p}_k \\ \boldsymbol{\theta}_k\end{bmatrix}$	$\begin{bmatrix}\boldsymbol{p}_2 - \boldsymbol{p}_1 \\ \boldsymbol{\theta}_2 - \boldsymbol{\theta}_1\end{bmatrix}$	$\begin{bmatrix}\boldsymbol{p} + \dot{\boldsymbol{p}}h \\ \boldsymbol{\theta} + \boldsymbol{W}(\boldsymbol{\theta})\boldsymbol{\omega}_b h\end{bmatrix}$

Representations choices: When using a different floating-base space representation we are faced with the following three main options/decisions: 1) how to select the optimization variables, 2) how to compute differences (e.g. for enforcing integration constraints or costs), and 3) how to perform the integration. We detail most of the possible combinations in Table I[5]: *we focus on the representation of the floating-base*, since we assume that the joint space is a vector space; in other words, we show only the variables related to the floating-base pose representation.

Implementation details: We use semi-implicit Euler integration for all spaces, and rely on the Pinocchio [19]

[3]For simplicity, we assume that the joints operate in a vector space, although this is not necessarily the case (e.g. a spherical joint). We can adapt the formulations using composite manifolds [10].

[4]We use multiple contact points per contact frame when necessary (e.g. we use 4 contact points for rectangular feet).

[5]We denote as $\text{Exp}_q(\cdot)$ the operation that transforms a translation vector and a quaternion to the SE(3) manifold [10]. $\text{Log}_q(\cdot)$ is the inverse operation.

and CasADi [20] libraries for acquiring gradients. For the SE(3) tangent representation we implement all the analytic gradients using the tools provided by Pinocchio. In particular, Pinocchio gives us the Jacobians on the SE(3) manifold, and we apply the appropriate chain rule to compute the Jacobians for $\mathfrak{se}(3)$. For the Quaternion-based and the Euler Angles representations we utilize the automatic differentiation provided by the CasADi-Pinocchio integration. Finally, as the optimization algorithm, we use the primal-dual interior point method as implemented in the Ipopt solver [13]. The code for the SE(3)-based trajectory optimization and the relevant experiments is available at `https://github.com/upatras-lar/se3_trajopt`.

V. Agile Movement Experiments

A. Experimental Scenarios

We evaluate five floating-base representations (see Tab. I) across a suite of agile whole-body tasks using three robots: the PAL Robotics Talos humanoid [21], the Unitree G1 humanoid, and the Unitree Go2 quadruped. Specifically, we test each representation on six motion scenarios, for a total of **30 optimization runs**.

Tasks: (see also Fig. 1)

- **Talos**:
  1) **Walk**: A Talos humanoid begins from a neutral standing pose and executes a straight-line walking gait to translate its base $2\,\mathrm{m}$ forward. The contact sequence follows a standard cycle (left-single support, right-single support, left-single support, right-single support, ...).
  2) **Hopscotch**: The Talos humanoid performs a hopscotch pattern over a $2\,\mathrm{m}$ course by alternately hopping twice on one foot. Each step consists of a single-support contact: first the left foot, then the left foot again, then the right foot, and finally the right foot again before coming to rest in a double support phase.
  3) **Big jump**: From a stationary stance, Talos jumps forward and lands precisely $1\,\mathrm{m}$ forward. The motion is split into a pre-launch stance phase, an aerial phase with no contacts, and a two-foot landing phase.
  4) **Handstand**: Talos transitions from standing to a handstand and holds the inverted pose in place. The trajectory comprises a crouch-to-handstand transfer phase and a phase with both hands in contact before "coming down" to a double foot support face again.
- **Unitree G1**:
  5) **Back-flip**: The Unitree G1 humanoid executes a full backward somersault, landing $0.5\,\mathrm{m}$ behind its takeoff point. The maneuver includes a preparatory phase that ends with a takeoff push, aerial rotation, and impact absorption upon landing.
- **Unitree Go2**:
  6) **Side-flip**: A Unitree Go2 quadruped performs a lateral somersault, departing from a neutral stance and landing $0.3\,\mathrm{m}$ sideways. The motion is divided into a preparatory shift, simultaneous hind- and fore-leg launch, aerial flip, and four-leg landing.

In Table II, we detail the cost functions and their weights per task, while we also report the discretization timestep and total duration of the behavior. The *Configuration Cost* refers to a regularization cost penalizing the configuration $\boldsymbol{q}_k$ from a reference $\boldsymbol{q}_0$. Similarly, the *Acceleration Cost* refers to a regularization cost penalizing big accelerations: that is, deviations of $\dot{\boldsymbol{v}}_k$ from $\mathbf{0}$.

As evaluation metrics for this scenario, we report the final cost (if any), number of iterations, number of objective function evaluations, number of Jacobian evaluations (for both the constraints and the objective function), and whether the optimization converged to a solution that solves the task.

TABLE II: Task Parameters.

Task	Configuration Cost	Acceleration Cost	Timestep	Duration
Walk (Talos)	none	none	$0.05\,s$	$3.3\,s$
Hopscotch (Talos)	none	none	$0.05\,s$	$4.3\,s$
Big Jump (Talos)	none	none	$0.05\,s$	$2.3\,s$
Handstand (Talos)	1×10^{-9}	none	$0.05\,s$	$3.4\,s$
Back-flip (G1)	1×10^{-9}	1×10^{-6}	$0.05\,s$	$2.4\,s$
Side-flip (Go2)	none	none	$0.02\,s$	$2.4\,s$

We also assess solver robustness on tasks 1, 3, 5 and 6 by adding zero-mean Gaussian noise of varying standard deviations to the initial warm-start trajectory. *We run 10 replicates* of each variation tuple for **a total 640 optimization runs**; we have 4 different tasks, 4 different representations (we ignore the "Quaternion #3' representation since it was not able to find solutions), and 4 different noise levels. *As an evaluation metric for this scenario,* we report success rates over the 10 replicates and noise levels, and the number of iterations to convergence.

Design criteria. An ideal trajectory optimizer should require minimal tuning per task or per robot. To this end, across all experiments we:

- Use only small regularization costs, with at most minor adjustments between tasks and platforms (see Table II);
- Initialize every node in a neutral stance (**non-informative warm-start**);
- Avoid robot-specific adaptations to the cost or solver settings;
- For the back-flip on G1, we introduce a slightly more informative warm-start: we set the floating-base orientation to the upside-down pose for a few nodes in the middle (in time) of the trajectory. Since we do not use any task-specific cost, this guides the optimizer toward a full somersault rather than a simple backward jump.

B. Experimental Results

The results reveal several key insights:

- **Uniform convergence on simple tasks.** For tasks that do not require large orientation changes (Tasks 1–4), all representations —except for "Quaternion #3"— converge to comparable trajectories in similar numbers of iterations and function evaluations (see Table IV).

- **Quaternion–manifold hybrids underperform.** The pure quaternion formulation ("Quaternion #1") outperforms both mixed quaternion–manifold variants ("Quaternion #2" and "Quaternion #3"), and "Quaternion #3" fails to converge in any experiment (see Table IV).
- **$\mathrm{SE}(3)$ Tangent-space excels on highly dynamic tasks.** For tasks involving significant reorientation (Tasks 5 and 6), the $\mathrm{SE}(3)$ tangent-space representation clearly outperforms all alternatives in solution quality (see Table IV). In particular, in the back- and side- flip tasks, the $\mathrm{SE}(3)$ tangent-space representation is able to converge to highly dynamic behaviors that perform the flips even though the warm-start is mildly informative. On the other hand, quaternion-based representations fail to converge to flipping solutions, and find simple jumping ones.
- **Robustness to initialization noise.** When we add zero-mean Gaussian noise of increasing magnitude to the initial guess, the $\mathrm{SE}(3)$ tangent-space parameterization consistently recovers agile motions (see Table III, Fig. 2). In this experiment, we measure success rate as the ability of the optimizer to converge to a feasible solution, and not the ability to solve the task (we reported this in Table IV). For example, in Task 5 and for noise level of $\sigma = 10^{-6}$, the "Quaternion #1" variant was converging to a simple jump behavior instead of the flip behaviors, but we report 100% success rate since all runs converged to the jumping behavior.

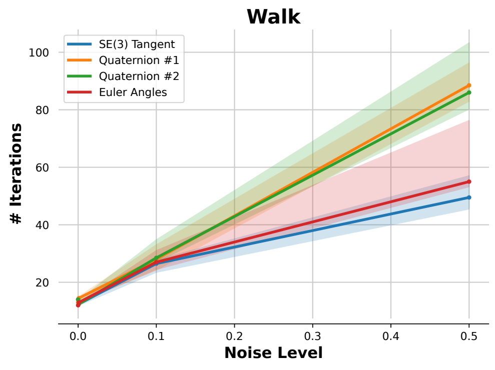
Overall, the $\mathrm{SE}(3)$ tangent-space representation matches or exceeds the performance of all other floating-base parameterizations. Nevertheless, "Quaternion #1' is a competitive alternative if one does not need very agile and dynamic movements or large re-orientations.

Qualitative analysis. Fig. 1 shows a few behaviors generated by the $\mathrm{SE}(3)$ Tangent space representation. In the *supplementary video* (also available at `https://lar.upatras.gr/projects/ibrics.html`), you can find more informative visualizations of the generated behaviors.

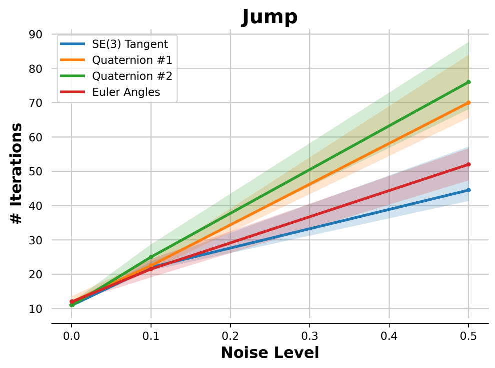
VI. Conclusion

In this work, we presented a systematic comparison of five floating-base parameterizations, based on Euler angles, quaternions, and the $\mathrm{SE}(3)$ tangent-space, in the context of agile whole-body motion planning for legged robots. Using identical transcription settings, solver parameters, and non-informative warm starts across six benchmark tasks, our experiments demonstrated that:

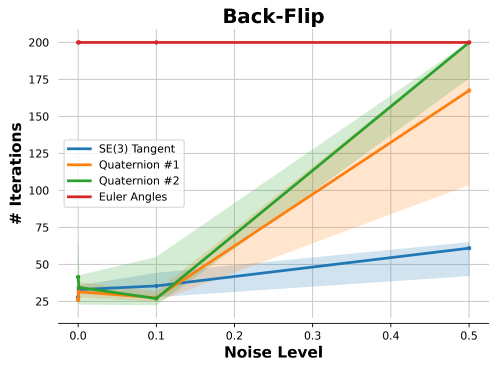
- All vector representations (Euler, Quaternion #1) perform similarly on low-rotation tasks (Task 1-4), but require more careful handling in high-rotation tasks (Task 5-6): Euler can fail to converge, while Quaternions might converge to suboptimal solutions.
- Quaternion–manifold hybrids underperform or even fail to converge.
- The $\mathrm{SE}(3)$ tangent-space formulation consistently yields the highest solution quality on dynamic movements that



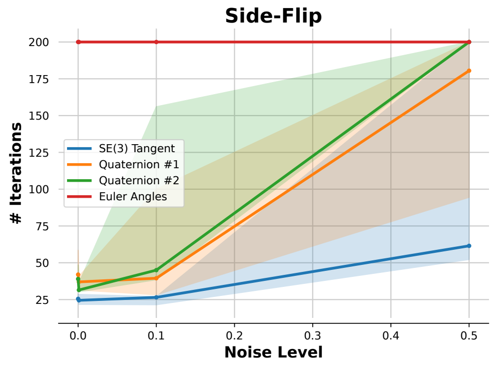
(a) Walk task



(b) Jump task



(c) Back-flip task



(d) Side-flip task

Fig. 2: Iterations to convergence versus noise levels. Solid lines are the median over 10 replicates and the shaded regions are the regions between the 25-th and 75-th percentiles. The $\mathrm{SE}(3)$ tangent representation shows the biggest robustness.

TABLE III: Success rates under varying warm-start noise levels.

Task	Representation	$\sigma = 10^{-6}$	$\sigma = 10^{-3}$	$\sigma = 0.1$	$\sigma = 0.5$
Walk (Talos)	Euler angles	100%	100%	100%	100%
	Quaternion #1	100%	100%	100%	100%
	Quaternion #2	100%	100%	100%	100%
	SE(3) Tangent	100%	100%	100%	100%
Big jump (Talos)	Euler angles	100%	100%	100%	100%
	Quaternion #1	100%	100%	100%	100%
	Quaternion #2	100%	100%	100%	90%
	SE(3) Tangent	100%	100%	100%	100%
Back-flip (G1)	Euler angles	0%	20%*	0%	0%
	Quaternion #1	100%*	100%*	100%*	60%*
	Quaternion #2	100%*	100%*	100%*	40%*
	SE(3) Tangent	100%	100%	100%	100%
Side-flip (Go2)	Euler angles	0%	0%	0%	0%
	Quaternion #1	100%*	100%*	80%*	50%*
	Quaternion #2	100%*	100%*	80%*	40%*
	SE(3) Tangent	100%	100%	100%	60%

* *Converged but did not find the flip behavior.*

require large re-orientations.

- SE(3) tangent-space is robust to perturbations in the initial guess, recovering agile behaviors under noisy initializations.

Overall, the SE(3) tangent-space representation provides the best balance between expressive power and compatibility with mature NLP solvers (e.g., Ipopt), enabling fast, reliable trajectory optimization for challenging legged-robot tasks.

Building on the promising performance of the SE(3) tangent-space formulation, our next step is to validate whether the produced behaviors can run on physical hardware. We are confident that this is possible given that simpler models have been deployed in real systems [2], [22]. In parallel, we plan to implement SE(3)-based Sequential Quadratic Programming (SQP) solvers dedicated to optimal control problems, which will allow us to run fast real-time Model Predictive Control (MPC) schemes (e.g. based on recent ADMM-based methods [23]); we are also planning to incorporate learned dynamics models [24], [25] to facilitate real-world deployment.

Finally, we aim to incorporate learning-based methods — e.g., leveraging demonstration data [26] or reinforcement-learning policies for better warm-starting [27], [28] or the other way around [29]— to reduce solve times, and generalize our approach beyond legged robots to other systems, such as multi-link manipulators and aerial manipulation platforms.

References

[1] T. Koolen, S. Bertrand, J. Thomas, S. Feng, M. Johnson, S. Kuindersma, and R. Tedrake, "Design of a momentum-based controller for humanoid robots with series elastic actuators," *Autonomous Robots*, vol. 40, no. 3, pp. 537–556, 2016.

[2] A. W. Winkler, D. G. Bellicoso, M. Hutter, and J. Buchli, "Gait and trajectory optimization for legged systems through phase-based end-effector parameterization," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1–9, IEEE, 2018.

[3] I. Tsikelis and K. Chatzilygeroudis, "Gait optimization for legged systems through mixed distribution cross-entropy optimization," in *2024 IEEE-RAS 23rd International Conference on Humanoid Robots (Humanoids)*, pp. 1011–1018, IEEE, 2024.

[4] M. Posa, C. Cantu, and R. Tedrake, "Direct trajectory optimization of rigid body dynamical systems through contact," in *Algorithmic foundations of robotics X*, pp. 527–542, Springer, 2014.

[5] S. Kuindersma, M. Fallon, H. Dai, F. Permenter, T. Koolen, P. Marion, and R. Tedrake, "Optimization-based locomotion planning, estimation, and control design for the atlas humanoid robot," *Autonomous Robots*, vol. 40, no. 3, pp. 429–455, 2016.

[6] P. M. Wensing, M. Posa, Y. Hu, A. Escande, N. Mansard, and A. Del Prete, "Optimization-based control for dynamic legged robots," *IEEE Transactions on Robotics*, vol. 40, pp. 43–63, 2023.

[7] B. E. Jackson, K. Tracy, and Z. Manchester, "Planning with attitude," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 5658–5664, 2021.

[8] W. Jallet, A. Bambade, E. Arlaud, S. El-Kazdadi, N. Mansard, and J. Carpentier, "Proxddp: Proximal constrained trajectory optimization," *IEEE Transactions on Robotics*, vol. 41, pp. 2605–2624, 2025.

[9] T. D. Barfoot, *State estimation for robotics*. Cambridge University Press, 2024.

[10] J. Solà, J. Deray, and D. Atchuthan, "A micro lie theory for state estimation in robotics," *CoRR*, vol. abs/1812.01537, 2018.

[11] I. Tsikelis, E. Tsiatsianas, C. Kiourt, S. Ivaldi, K. Chatzilygeroudis, and E. Mingo Hoffman, "Multi-Contact Agile Whole-Body Motion Planning via Contact Sequence Discovery and SE(3) Tangent-Space Trajectory Optimization," in *2025 IEEE-RAS 24th International Conference on Humanoid Robots (Humanoids)*, 2025.

[12] S. Teng, T.-Y. Lin, W. A. Clark, R. Vasudevan, and M. Ghaffari, "Riemannian Direct Trajectory Optimization of Rigid Bodies on Matrix Lie Groups," in *Robotics: Science and Systems (RSS)*, 2025.

[13] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical Programming*, vol. 106, no. 1, pp. 25–57, 2006.

[14] P.-A. Absil, R. Mahony, and R. Sepulchre, *Optimization algorithms on matrix manifolds*. Princeton University Press, 2008.

[15] H. Ferrolho, V. Ivan, W. Merkt, I. Havoutis, and S. Vijayakumar, "Inverse dynamics vs. forward dynamics in direct transcription formulations for trajectory optimization," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 12752–12758, IEEE, 2021.

[16] R. Featherstone, *Rigid Body Dynamics Algorithms*. Springer, 2014.

[17] K. M. Lynch and F. C. Park, *Modern robotics*. Cambridge University Press, 2017.

[18] K. I. Chatzilygeroudis, C. G. Tsakonas, and M. N. Vrahatis, "Evolving dynamic locomotion policies in minutes," in *2023 14th International Conference on Information, Intelligence, Systems & Applications (IISA)*, pp. 1–8, IEEE, 2023.

[19] J. Carpentier, G. Saurel, G. Buondonno, J. Mirabel, F. Lamiraux, O. Stasse, and N. Mansard, "Pinocchio: Fast forward and inverse dynamics for poly-articulated systems," *HAL Open Science*, 2022.

[20] J. A. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "Casadi: a software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, vol. 11, pp. 1–36, 2019.

[21] O. Stasse, T. Flayols, R. Budhiraja, K. Giraud-Esclasse, J. Carpentier, J. Mirabel, A. Del Prete, P. Souères, N. Mansard, F. Lamiraux, *et al.*, "Talos: A new humanoid research platform targeted for industrial applications," in *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*, pp. 689–695, IEEE, 2017.

[22] Y. Ding, A. Pandala, C. Li, Y.-H. Shin, and H.-W. Park, "Representation-free model predictive control for dynamic motions in quadrupeds," *IEEE Transactions on Robotics*, vol. 37, no. 4, pp. 1154–1171, 2021.

[23] A. Jordana, S. Kleff, A. Meduri, J. Carpentier, N. Mansard, and L. Righetti, "Stagewise implementations of sequential quadratic programming for model-predictive control," *Subm. IEEE TRO*, 2023.

[24] F. Khadivar, K. Chatzilygeroudis, and A. Billard, "Self-correcting quadratic programming-based robot control," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 53, no. 8, pp. 5236–5247, 2023.

[25] K. Chatzilygeroudis, R. Rama, R. Kaushik, D. Goepp, V. Vassiliades, and J.-B. Mouret, "Black-box data-efficient policy search for robotics,"

TABLE IV: Optimization results for five floating-base representations across six tasks.

Task	Representation	Cost	Iterations	Obj. evals	Jac. evals	Success
Walking (Talos)	Euler angles	–	12	13	42	Yes
	Quaternion #1	–	13	15	42	Yes
	Quaternion #2	–	14	16	45	Yes
	Quaternion #3	–	–	–	–	No
	SE(3) Tangent	–	14	16	45	Yes
Hopscotch (Talos)	Euler angles	–	14	18	45	Yes
	Quaternion #1	–	12	13	39	Yes
	Quaternion #2	–	16	17	51	Yes
	Quaternion #3	–	–	–	–	No
	SE(3) Tangent	–	10	14	33	Yes
Big jump (Talos)	Euler angles	–	12	13	39	Yes
	Quaternion #1	–	11	12	36	Yes
	Quaternion #2	–	11	12	36	Yes
	Quaternion #3	–	–	–	–	No
	SE(3) Tangent	–	11	12	36	Yes
Handstand (Talos)	Euler angles	3.710e-7	53	68	162	Yes
	Quaternion #1	3.71e-7	58	58	135	Yes
	Quaternion #2	3.81e-7	96	96	159	Yes
	Quaternion #3	–	–	–	–	No
	SE(3) Tangent	3.91e-7	88	104	267	Yes
Back-flip (G1)	Euler angles	12.5	200	414	603	No
	Quaternion #1	0.1	26	30	81	No*
	Quaternion #2	0.1	71	84	216	No*
	Quaternion #3	–	–	–	–	No
	SE(3) Tangent	5.2e-5	28	47	87	Yes
Side-flip (Go2)	Euler angles	–	200	632	603	No
	Quaternion #1	–	39	45	120	No*
	Quaternion #2	–	56	316	171	No*
	Quaternion #3	–	–	–	–	No
	SE(3) Tangent	–	29	50	90	Yes

* *The solver converged to a feasible solution, but it was not performing the flip. See also the supplementary video.*

in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 51–58, IEEE, 2017.

[26] D. Totsila, K. Chatzilygeroudis, V. Modugno, D. Hadjivelichkov, and D. Kanoulas, “Sensorimotor Learning With Stability Guarantees via Autonomous Neural Dynamic Policies,” *IEEE Robotics and Automation Letters*, 2025.

[27] O. Melon, M. Geisert, D. Surovik, I. Havoutis, and M. Fallon, “Reliable trajectories for dynamic quadrupeds using analytical costs and learned initializations,” in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1410–1416, IEEE, 2020.

[28] K. Chatzilygeroudis, V. Vassiliades, F. Stulp, S. Calinon, and J.-B. Mouret, “A survey on policy search algorithms for learning robot controllers in a handful of trials,” *IEEE Transactions on Robotics*, vol. 36, no. 2, pp. 328–347, 2019.

[29] M. Bogdanovic, M. Khadiv, and L. Righetti, “Model-free reinforcement learning for robust locomotion using demonstrations from trajectory optimization,” *Frontiers in Robotics and AI*, vol. 9, p. 854212, 2022.